A Robust Boosting Method using Zero-one Loss Function :SNRBoost (6th Workshop on Stochastic Numerics)

AUTHOR(S):

Sano, Natsuki; Suzuki, Hideo; Koda, Masato

数理解析研究所講究録 1351 巻 2004 年 106-121

A Robust Boosting Method using Zero-one Loss Function : SNRBoost

筑波大学大学院・社会工学研究科　佐野　夏樹 (Natsuki Sano)
The Doctoral Program in Policy and Planning Sciences
University of Tsukuba
筑波大学・社会工学系　鈴木　秀男 (Hideo Suzuki)
Institute of Policy and Planning Sciences
University of Tsukuba
筑波大学・社会工学系　香田　正人 (Masato Koda)
Institute of Policy and Planning Sciences
University of Tsukuba

Abstract

We propose a new, robust boosting method by using a zero-one step function as a loss function. In deriving the method, the margin boost technique is blended with the derivative approximation algorithm, called *Stochastic Noise Reaction* (SNR). Based on intensive numerical experiments, we show that the proposed method is actually better than other boosting methods on test error rates in the case of noisy, mislabeled situation.

Key words: AdaBoost, MarginBoost, Zero-one Loss Function, Stochastic Noise Reaction.

1. Introduction–AdaBoost

Suppose a situation in which a management must solve a decision problem depending on a number of people whose opinions differ slightly, and their experiences and personal abilities to solve the problem seem to be almost equal and, as a result, cannot draw a decisive conclusion. To resolve this situation in order to make a reasonably better decision, is it more efficient to ignore opinions of these people and relying solely on manager's decision, or to restart the discussion by changing the team and forming a new ensemble through addition of capable people?

Obviously, the first behaviour is faster but it may not lead to a better decision. On the other hand, the second one may lead to a better decision since it will provide an iterative improvement to the solution but it certainly is a slow process and takes much longer time. This is what we often encounter during decision analysis dealing with data mining. However, we can make a better decision even in the original situation by resorting to the iterative improvement method considering that a possible better solution is still the outcome of a combination of a set of slightly different opinions converging toward the genuine better one.

The situation described above is essentially a problem associated with a committee-based decision making. The aim of this paper is to develop a new robust algorithm

for pattern classification problem by using an analogy to the committee-based decision making, where each individual member of the committee corresponds to a classifier. The decision here is to correctly classify data to its true class label, and we would like to develop a new robust classification method through iterative improvement of classifiers or committee members.

In view of the above objective in mind, the starting point for the present study would be an iterative improvement procedure called *boosting*, which is a way of combining the performance of many *weak* classifiers to produce a powerful committee. The procedure allows the designer to continue adding weak classifiers until some desired low training error has been achieved. Boosting techniques have mostly been studied in the computational learning theory literature (e.g., see Schapire (1990), Freund (1995), Freund and Schapire (1997)) and received increasing attention in many areas including data mining and knowledge discovery.

While boosting has evolved over the recent years, we focus on the most commonly used version of the adaptive boosting procedure, i.e., " AdaBoost.M1. " (Freund and Schapire, 1997). A concise description of AdaBoost is given here for the two-category classification setting. We have a set of n trainning data pairs, $(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_n, y_n)$, where x_i denotes a vector valued feature with $y_i = -1$ or 1, as its class label or teacher signal. The total number of component classifiers is assumed to be T. Then, the output of classification model (i.e., committee) is given by a scalar function, $F(x) = \sum_{t=1}^{T} \beta_t f_t(x)$, in which each $f_t(x)$ denotes a classifier producing values ± 1, and β_t are prescribed constants; the corresponding prediction (i.e., decision) is $\text{sign}(F(x))$.

The AdaBoost procedure trains the classifiers $f_t(x)$ on weighted versions of the training sample, by giving higher weight to cases that are currently misclassified. This is done for a sequence of weighted samples, and then the final classifier is defined to be a linear superposition of the classifiers from each stage. A detailed desciption of AdaBoost.M1. is summarized and given in the next box, where $I(y_i \neq f_t(x_i))$ denotes the indicator function with respect to the event such that $y_i \neq f_t(x_i)$, i.e., misclassification event.

At t-th iteration stage, given the observation weight $w_{t-1}(i)$, AdaBoost fits a classifier $f_t(x)$ to the training data x_i $(i = 1, 2, \ldots, n)$ that is weighted by $w_{t-1}(i)$. Then, it evaluates the weighted error err_t by computing a weighted count of misclassification events as $\text{err}_t = \sum_{i=1}^{n} w_{t-1}(i) I(y_i \neq f_t(x_i))$. Using err_t, β_t is obtained as $\beta_t = \log((1 - \text{err}_t)/\text{err}_t)$. Finally, the observation weight is updated through the computation of $w_t(i) = w_{t-1}(i) \exp[\beta_t I(y_i \neq f_t(x_i))]$ and normalized so that $\sum_{i=1}^{n} w_t(i) = 1$. This results in giving a heigher weight to the training data that is currently misclassified.

Much has been written about the success of AdaBoost in producing accurate classifiers. Many authors have explored the use of a tree-based classifier for $f_t(x)$ and demonstrated that it consistently produces significantly lower error rates than a single decision tree. In fact, Breiman called AdaBoost with trees as " the best off-the-shelf classifier in the world (Breiman, 1998). "

AdaBoost.M1. (Freund and Schapire, 1997)

1. Initialize the observation weights $w_0(i) = 1/n$ for $i = 1, 2, \ldots, n$, and $F_0(x) = 0$.
2. For $t = 1, 2, \ldots, T$ do
   (a) Fit a classifier $f_t(x)$ to the training data weighted by $w_{t-1}(i)$.
   (b) Compute $\mathrm{err}_t = \sum_{i=1}^{n} w_{t-1}(i) I(y_i \neq f_t(x_i))$.
   (c) Compute $\beta_t = \log((1 - \mathrm{err}_t)/(\mathrm{err}_t))$.
   (d) Let $F_t(x) = F_{t-1}(x) + \beta_t f_t(x)$.
   (e) Set $w_t(i) = w_{t-1}(i) \exp[\beta_t I(y_i \neq f_t(x_i))]$, and normalize $w_t(i)$ so that $\sum_{i=1}^{n} w_t(i) = 1$ for $i = 1, 2, \ldots, n$.

   end For
3. Output $\mathrm{sign}(F_T(x))$.

Interestingly, in many examples, the test error seems to consistently decrease and then level off as more classifiers are added, instead of turning into ultimate increase. It hence seems that AdaBoost is resistent to overfitting for low noise cases. However, recent studies with highly noisy patterns (Quinlan, 1996; Grove and Schuurmans, 1998; Rätsch et al., 1998) depict that it is clearly a myth that AdaBoost do not overfit since AdaBoost asymptotically concentrate on the patterns which are hardest to learn. To cope with this problem, some regularized boosting algorithms (Mason et al., 1999; Rätsch et al., 1998) are proposed. In this paper, we will take an iterative improvement approach to optimize classifiers to derive a new robust boosting method that is resistent against mislabeled noisy patterns.

In the next section, we present the principles of MarginBoost. We also roughly explain an outline of LogitBoost in Section 2. Then, in Section 3, we present the zero-one loss function and Stochastic Noise Reaction (SNR). In Section 4, we propose the new robust boosting method and results of intensive numerical experiments are presented, and we detail cases with noisy, mislabeled patterns. Section 5 contains some concluding remarks.

2. Margin Boost

We assume that the training data set (x, y) is randomly generated according to some unknown probability distribution $\mathcal{D}$ on $X \times Y$ where X is the space of measurements (typically $X \subseteq \mathbb{R}^N$) and Y is the space of labels (Y is usually a discrete set or some subset of $\mathbb{R}$). The output of classification is denoted as $\mathrm{sign}(F(x))$, where $F(x)$ is

given by

$$F(x) = \sum_{t=1}^{T} \beta_t f_t(x), \tag{1}$$

and $f_t(x) : X \to \{\pm 1\}$ are base classifiers from some fixed class $\mathcal{F}$, and $\beta_t \in \mathbb{R}$ denotes the classifier weights. The *margin* of the training data (x, y) with respect to the classifier, i.e., $\text{sign}(F(x))$, is defined as $yF(x)$. It is noted that if $y \neq \text{sign}(F(x))$ then $yF(x) < 0$, and if $y = \text{sign}(F(x))$ then $yF(x) \geq 0$. Margin can be understood as a mesure of difficulty of classification. As data has a larger (positive) margin, the example is easier to classify. On the contrary, as it has a smaller (negative) margin, the data is harder to classify. Given a set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of n labelled data pairs generated according to $\mathcal{D}$, we wish to construct a classification model described by Equation (1) so that the probability of incorrect classification $P_{\mathcal{D}}(\text{sign}(F(x)) \neq y)$ is small. Since $\mathcal{D}$ is unknown and we are only given a training set S, we take the approach of finding the classification model which minimizes the sample risk of some cost function of the margin. That is, for a training set S we want to find F such that the empirical average,

$$L(F) = \frac{1}{n} \sum_{i=1}^{n} C(y_i F(x_i)) \tag{2}$$

is minimized for some suitable cost function $C : \mathbb{R} \to \mathbb{R}$. The interpretation of AdaBoost as an algorithm which performs a gradient descent optimization of the sample risk has been examined by several authors, e.g., Friedman et al. (2000), and Mason et al. (2000).

In the next box, MarginBoost is summarized, which gives a general framework of AdaBoost in terms of the margin. AdaBoost can be seen as a special case of MarginBoost in case of the cost function defined by $C(z) = e^{-z}$, where z denotes the margin, $z = yF(x)$. In the box, $C'(z)$ denotes the derivative of the margin cost function with respect to z. Note that $C'(z)$ is nonpositive since the cost function is monotonically decreasing and the normalized weight $w_{t+1}(i) = \frac{C'(y_i F_{t+1}(x_i))}{\sum_{i=1}^{n} C'(y_i F_{t+1}(x_i))} > 0$ can be interpreted as a probability, but it actually gives a gradient information. MarginBoost, hence, falls into the category of gradient-type search algorithm. If $\sum_{i=1}^{n} w_t(i) y_i f_{t+1}(x_i) \geq 0$, then algorithm returns F_t and terminates. The readers are referred to Mason et al. (2000) for details of MarginBoost.

2.1. Logit Boost

Friedman et al. (2000) derived LogitBoost that fits additive logistic regression models by stagewise optimization of the binomial deviance loss. In the following box, LogitBoost is summarized. In the box, u_i is a response value, and y_i^* denotes response such that $y_i^* \in \{0, 1\}$, which is gained by $y^* = (y + 1)/2$. They set a weak learner by means of the least-square regression. The readers are referred to Friedman

MarginBoost (Mason et al., 2000)

0. Specify a suitable cost function C.
1. Initialize $w_0(i) = 1/n$ for $i = 1, 2, \ldots, n$, and $F_0(x) = 0$.
2. For t = 1,2,...,T, do
   (a) Fit a classifier $f_t(x)$ to the training data weighted by $w_{t-1}(i)$ for $i = 1, 2, \ldots, n$.
   (b) If $\sum_{i=1}^{n} w_{t-1}(i) y_i f_t(x_i) \geq 0$, then return F_t.
   end If.
   (c) Choose β_t appropriately.
   (d) Let $F_t(x) = F_{t-1}(x) + \beta_t f_t(x)$.
   (e) Set $w_t(i) = \frac{C'(y_i F_t(x_i))}{\sum_{i=1}^{n} C'(y_i F_t(x_i))}$ for $i = 1, 2, \ldots, n$.
   end For
3. Output $\text{sign}(F_T(x))$.

LogitBoost (Friedman et al., 2000)

1. Initialize the observation weights $w_i = 1/n$ for $i = 1, 2, \ldots, n$, and $F(x) = 0$ and probability estimates $p(x_i) = \frac{1}{2}$.
2. For t=1,2,...,T, do
   (a) Compute the working response u_i and weight w_i

$$\begin{aligned} u_i &= \frac{y_i^* - p(x_i)}{p(x_i)(1 - p(x_i))} \\ w_i &= p(x_i)(1 - p(x_i)) \end{aligned}$$

   (b) Fit a classifier $f_t(x)$ by a weighted least-square regression of u_i to x_i using weight w_i.
   (c) Update

$$F(x) \leftarrow F(x) + \frac{1}{2} f_t(x), \text{ and } p(x) \leftarrow \frac{e^{F(x)}}{e^{F(x)} + e^{-F(x)}}.$$

   end For
3. Output $F_T(x) = \text{sign}[\sum_{t=1}^{T} f_t(x)]$.

et al. (2000) for details of the LogitBoost. In the present study, we use LogitBoost for numerical experiments, where a neural network is utilized as a weak learner.

3. Misclassification Loss Function and Stochastic Noise Reaction

3.1. Misclassification Loss Function

Friedman et al. (2000) have demonstrated that the AdaBoost algorithm decreases an exponential loss function. Figure 1 illustrates various loss functions as a function of the margin value, $z = yF(x)$, including the exponential loss function. When all the class labels are not mislabeled and hence data is error-free, the result of correct classification always yields a positive margin since y and $F(x)$ both share the same sign while incorrect one yields negative margin. The loss function for Support Vector Machine is also shown in Figure 1, which is a statistical learning method to train kernel-based machines with optimal margins by mapping training data in a higher dimension.

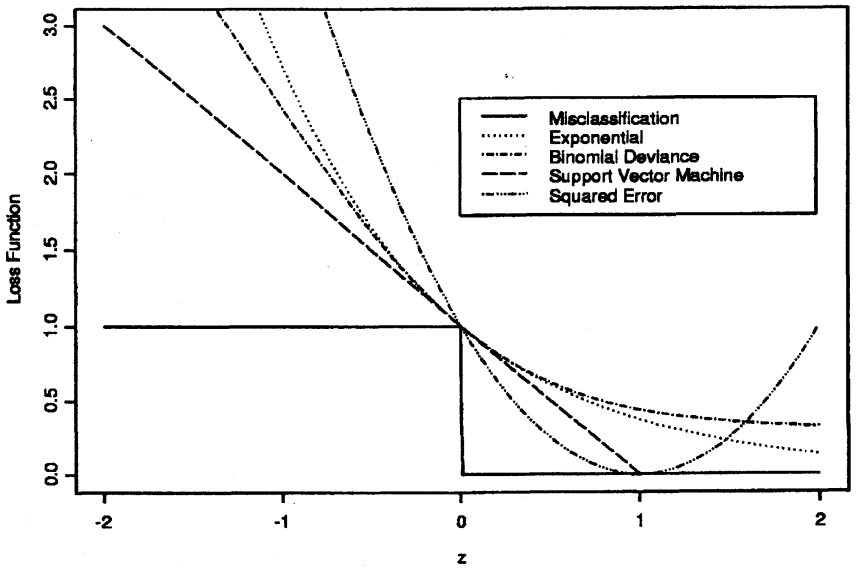


Figure 1: Loss functions for two-category classification

Shown also in Figure 1 is the misclassification loss (zero-one loss function), $C(z) = I(yF(x) < 0)$, where $I(yF(x) < 0)$ denotes the indicator function with respect to the occurence of the incorrect events, i.e., $yF(x) < 0$, which gives unit penalty for negative margin values, with no penalty for positive ones (i.e., correct

decisions). Note that the misclassification loss is a discontinuous step function. In this way, the decision rule becomes a judgement on a zero-one loss function, which will be investigated in this study. It is important to note that the zero-one loss function yields the Bayes minimum classificaton error for binary classification (Duda et al., 2001).

The exponential loss function exponentially penalizes negative margin observations or incorrect decisions. At any point in the training process, the exponential criterion concentrates much more influence on observations with large negative margins. This is considered as one of the reasons why AdaBoost is not robust for noisy situation where there is misspecification of the class labels in the training data. Since the zero-one loss function concentrates and uniquely influences on negative margin, it is far more robust in noisy setting where the Bayes error rate is not close to zero, which is especially the case in mislabeled situation.

Mean-squared approximation errors are well-understood and used as a loss function in statistics area. Unlike the zero-one loss function which considers only the misclassified observations, the minimum squared error (MSE) criterion takes into account the entire training samples with wide range of margin values. Hence, if MSE is adopted, the correct classification but with $yF(x) > 1$ incurrs increasing loss for larger values of $|F(x)|$. This makes the squared-error a poor approximation compared to the zero-one loss function and not desirable since the classification results that are "excessively" correct are also penalized as much as worst (extremely incorrect) cases. Other functions in Figure 1 can be viewed as monotone continuous approximations to the misclassification loss. We note that LogitBoost uses the binomial deviance loss.

Here we would like to propose the zero-one loss function which takes account of limited influences from larger negative margins in a proper manner and, accordingly, robust against mislabeled noisy training data. Actual misclassification loss is not differentiable at the origin and therefore we use *Stochastic Noise Reaction* (SNR) technique proposed by Koda and Okano (2000) in order to approximate the derivative of the zero-one loss function.

3.2. Stochastic Noise Reaction

Optimization problems that seek for minimization (or equivalently maximization) of an objective function have practical importance in various areas. Once a task is modeled as an optimization problem, general optimization technques become applicable; e.g., linear programming, gradient methods, etc. One may encounter difficulties, however, in applying these techniques when the objective function is non-differentiable or when it is defined as a procedure. To deal with such a case, Koda and Okano (2000) proposed a function minimization algorithm, called Stochastic Noise Reaction (SNR) that updates solutions based on approximated derivative information. They also apply SNR to *traveling salesman problem* (TSP), a representative combinatorial optimization problem (Okano and Koda, 2003). We illustrate here only an essence of SNR needed in boosting.

A gradient vector of an objective function $f(x)$ is defined by

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \cdots, \frac{\partial f(x)}{\partial x_N} \right)^T. \tag{3}$$

To avoid the exact gradient calculation of Equation (3), SNR injects a Gaussian white noise, $\xi_i \in N(0,1)$, into a variable x_i as

$$x_i(j) = x_i + \xi_i(j), \tag{4}$$

where $\xi_i(j)$ denotes the j-th realization of the noise injected into the i-th variable. Each component of a derivative is approximated without explicitly defferentiating the objective function by using the relation

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{1}{M} \sum_{j=1}^{M} f(x(j))\xi_i(j), \tag{5}$$

where M is a loop count for taking the average. The value of M was set to 100 in all of the numerical experiments here. In actual implementations, all the realizations of the noise should be generated and normalized in advance to ensure $\langle \xi_i \rangle = 0$ and $\mathrm{var}(\xi_i) = 1$, where $\langle \cdot \rangle$ denotes the expectation operator.

3.3. Mathematical framework

The stochastic sensitivity analysis method, Equation (5), uses the following relationship:

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{1}{\sigma_i^2} \langle f(x+\xi)\xi_i \rangle, \tag{6}$$

where $f(x)$ is a continuously differentiable function, and ξ is an N-dimensional vector each of whose components ξ_i is an independent Gaussian noise with mean 0 and standard deviation σ_i; i.e.,

$$\langle \xi_i \xi_j \rangle = \sigma_i^2 \delta_{ij}, \quad \langle \xi_i^{2r+1} \rangle = 0, \quad \langle \xi_i^{2r} \rangle = \sigma_i^{2r} \frac{(2r)!}{2^r r!}, \tag{7}$$

where δ_{ij} denotes the Kronecker delta, and r is a non-negative integer value.

Using (7) in a Taylor series expansion of $f(x+\xi)$ around x gives

$$\begin{aligned}
\tfrac{1}{\sigma_i^2}\langle f(x+\xi)\xi_i \rangle &= \tfrac{1}{\sigma_i^2}\langle \{f(x) + \textstyle\sum_{k=1}^{\infty} \frac{1}{k!} \sum_{j=1}^{N} (\xi_j \frac{\partial}{\partial x_j})^k f(x)\}\xi_i \rangle \\
&= \tfrac{1}{\sigma_i^2}\{f(x)\langle \xi_i \rangle + \textstyle\sum_{k=1}^{\infty} \frac{1}{k!} \sum_{j=1}^{N} \langle \xi_i (\xi_j \frac{\partial}{\partial x_j})^k \rangle f(x)\} \\
&= \tfrac{1}{\sigma_i^2} \textstyle\sum_{k=1}^{\infty} \frac{1}{k!} \langle \xi_i^{k+1} \rangle \frac{\partial^k}{\partial x_i^k} f(x) \\
&= \tfrac{1}{\sigma_i^2} \textstyle\sum_{k=2,4,6,\ldots} \frac{1}{(k-1)!} \langle \xi_i^k \rangle \frac{\partial^{k-1}}{\partial x_i^{k-1}} f(x) \\
&= \tfrac{\langle \xi_i^2 \rangle}{\sigma_i^2} \tfrac{\partial f(x)}{\partial x_i} + \textstyle\sum_{k=4,6,8,\ldots} \frac{1}{(k-1)!} \frac{\langle \xi_i^k \rangle}{\sigma_i^2} \frac{\partial^{k-1}}{\partial x_i^{k-1}} f(x) \\
&= \tfrac{\partial f(x)}{\partial x_i} + \textstyle\sum_{r=2}^{\infty} \frac{1}{(r-1)!} \left(\frac{\sigma_i}{\sqrt{2}} \right)^{2(r-1)} \frac{\partial^{2r-1}}{\partial x_i^{2r-1}} f(x) \\
&\approx \tfrac{\partial f(x)}{\partial x_i},
\end{aligned} \tag{8}$$

for sufficiently small $\sigma_i < \sqrt{2}$. It is expected that Equation (8) yields a good approximation when $\sigma_i \to 0$. In Equation (5), for simplicity and clarity reasons, $\sigma_i = 1$ is assumed, and the expectation is replaced by the sample mean.

In Equation (8), it is important to note that the differential operator has disappeared and the gradient information is given as an expected value of the product of the objective function $f(x + \xi)$ and the noise ξ_i. Since the present method will smoothen the objective function to be estimated by sampling and averaging, it may be efficiently applied to discontinuous objective functions, such as piecewise constant functions or zero-one loss function. Based on Novikov's Theorem for functional derivatives (interested readers are referred to Novikov (1965)), a similar idea and methodology have been applied to the sensitivity analysis of stochastic kinetic models (Dacol and Rabitz, 1984) and the stochastic formulation of neural networks (Koda and Okano, 2000). Analogous sensitivity analysis techniques have recently been suggested for the computation of sensitivities (i.e., Greeks) in finance using the stochastic calculus of variations, known as Malliavin calculus (Fournie et al., 2001).

3.4. Application to Zero-one Loss Function

Based on Equation (5), the approximated derivative for misclassification loss, i.e., zero-one loss function, is shown in Figure 2. Figure 2 shows that the discontinuous point of the zero-one loss function at $x = 0$ is leveled, and the derivative is obtained. Note that the approximated gradient resembles the derivative of $C(x) = \frac{1}{2}\{-\tanh(x) + 1\}$, which is the smoothed zero-one loss function and is differentiable as $C'(x) = \frac{1}{2}(\tanh(x) + 1)(\tanh(x) - 1)$. It is recognized that the approximated derivative is negative and smallest at the origin, and equals to 0 as x is distant from the origin. It is difficult to observe, however, that there could be a positive derivative at distant points from the origin.

4. SNRBoost and Numerical Experiments

4.1. SNR Boost

The proposed boosting algorithm SNRBoost is illustrated. The margin of i-th training data at t-th iteration is denoted as $z_t(i) = y_i F_t(x_i)$. The approximated derivatives using SNR for the zero-one loss function at $z_t(i)$ is denoted by $d_t(i)$ for $i = 1, \ldots, n$. If the weight $w_t(i)$ is less than 0, it is replaced by 0 and the computing sequence is continued.

For the appropriate choice of β_t, we set $\beta_t = 1/t$, based on the stochastic approximation technique (Duda et al, 2001), which satisfies the conditions

$$\lim_{T\to\infty} \sum_{t=1}^{T} \beta_t = \infty, \text{ and } \lim_{T\to\infty} \sum_{t=1}^{T} \beta_t^2 < \infty. \tag{9}$$

The proposed method of β_t guarantees a convergence of F_t if an infinite sequence is applied to the gradient-type search method, which is the present situation.

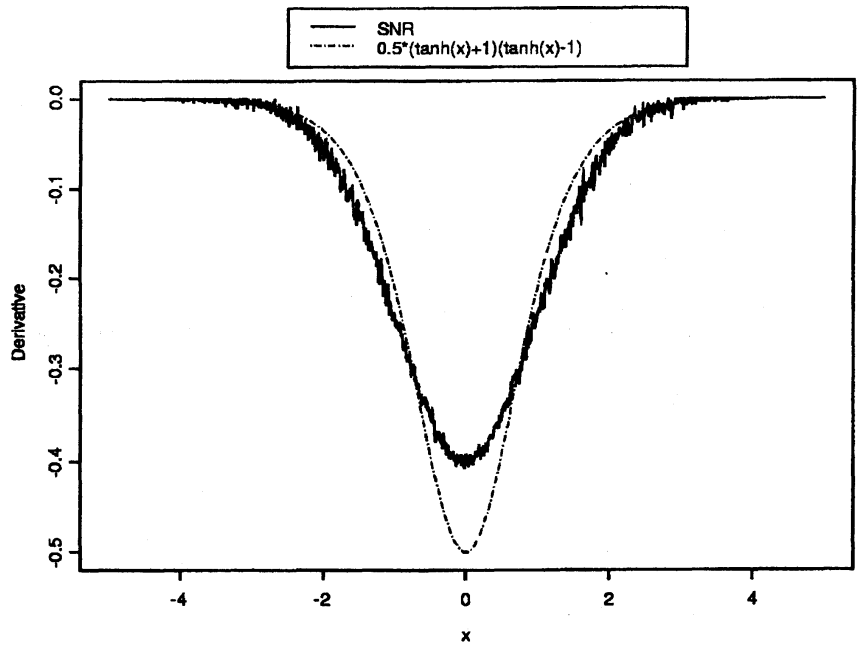


Figure 2: Derivative approximation for zero-one loss function

SNRBoost

1. Initialize $w_0(i) = 1/n$ for $i = 1, 2, \ldots, n$, and $F_0(\mathbf{x}) = 0$.
2. For $t = 1, 2, \ldots, T$, do
   (a) Fit a classifier $f_t(x)$ to the training data weighted by $w_{t-1}(i)$ for $i = 1, 2, \ldots, n$.
   (b) Set $\beta_t = 1/t$.
   (c) $F_t(x) = F_{t-1}(x) + \beta_t f_t(x)$.
   (d) Compute $z_t(i) = y_i F_t(x_i)$.
   (e) Compute approximated derivatives for zero-one loss function $d_t(i)$ at $z_t(i)$ using SNR for $i = 1, 2, \ldots, n$.
   (f) Compute weights $w_t(i) = \frac{d_t(i)}{\sum_{i=1}^{n} d_t(i)}$ for $i = 1, 2, \ldots, n$.
   (g) If $w_t(i) \leq 0$ then $w_t(i) = 0$.
   end If.

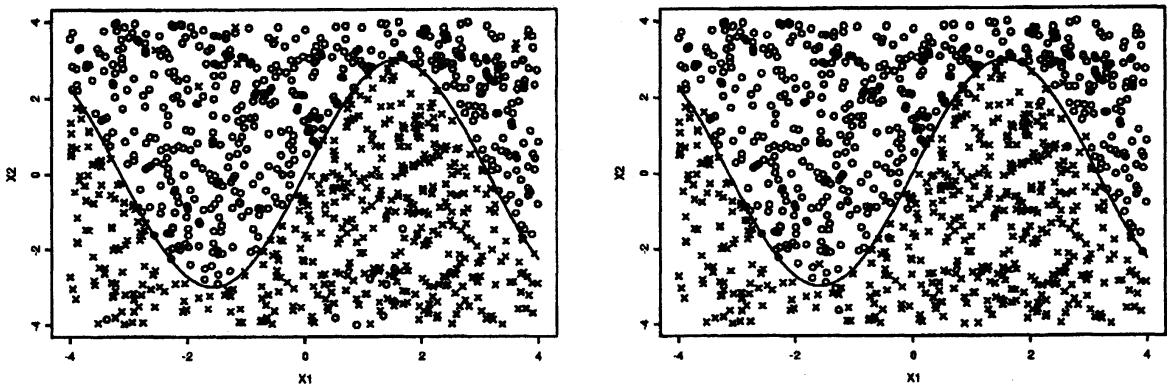
   end For
3. Output sign$(F_T(x))$.

4.2. Numerical Experiments

In this section, numerical results are presented and, especially, the robustness of the proposed method are analyzed and compared with that of AdaBoost and LogitBoost. We focus our attention to classification results for mislabeled (i.e., noisy) cases. The back-propagation neural network with single hidden layer is used as a base learner for all the three boosting methods. The number of units in hidden layer (i.e., hidden units) is 3. Since a multilayer neural network has been shown to be able to define an arbitrary decision function, with a flexible architecture in terms of the number of hidden units, it thus offers the potential of ideal base learner for the experiments.

For numerical experiments, data (with 2% mislabeled case) is generated as follows:

1. Generate uniformly $\mathbf{x} = (x_1, x_2) \in \chi = [-4, 4] \times [-4, 4]$;
2. assign $y = \text{sign}(F(\mathbf{x}))$, where $F(\mathbf{x}) = x_2 - 3\sin(x_1)$;
3. sort $|F(\mathbf{x}_i)|$ by descending order;
4. sample randomly 2% from top 20% (far case) or lowest 20% (near case) examples;
5. flip sampled examples in Step 4.



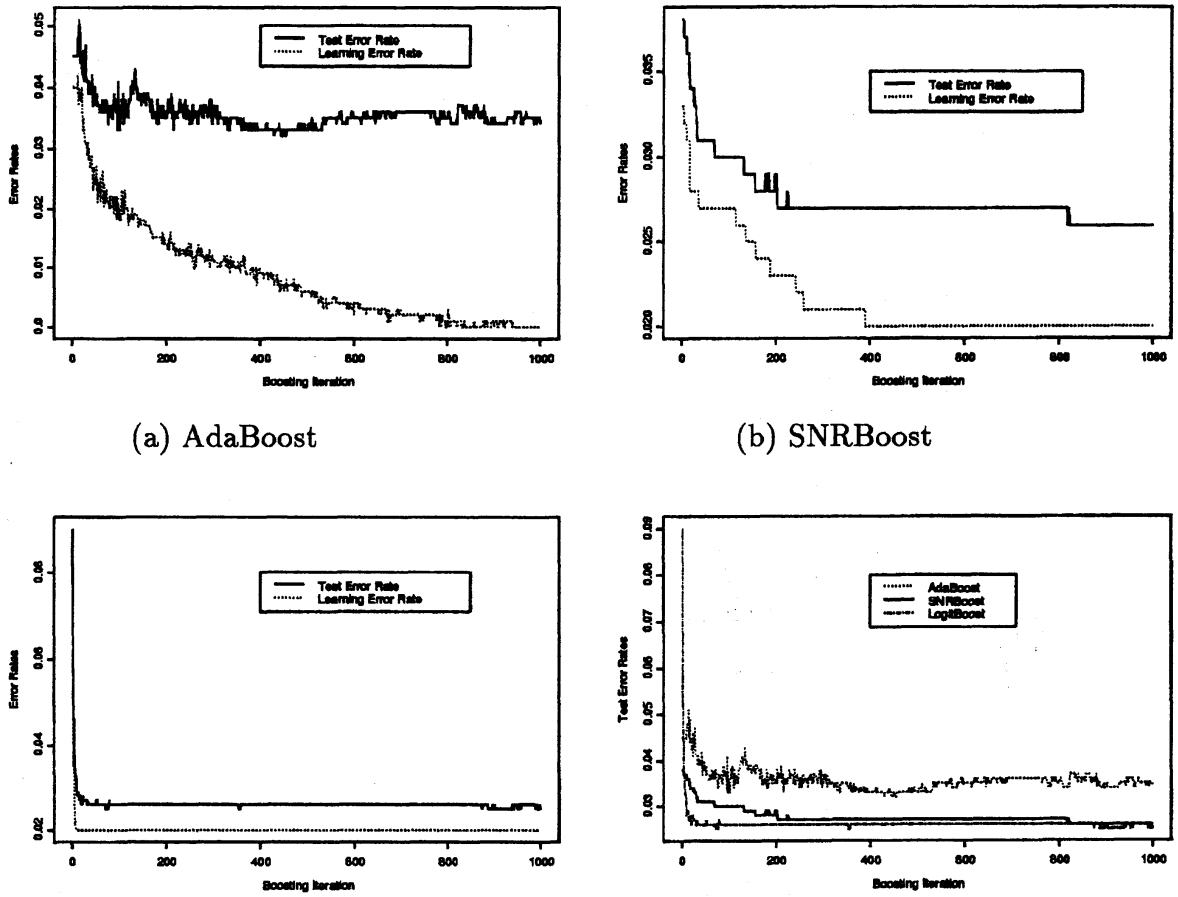
(a) Far Case (b) Near Case

Figure 3: Learning Data. (a) Case of 2% mislabed data located far from the decision boundary. (b) Case of 2% mislabed data concentrated near the decision boundary.

In Step 2, note that $F(\mathbf{x}) = x_2 - 3\sin(x_1)$ is used as a nonlinear decision function. In Step 4, we generate two mislabeled cases; one where mislabeled data are located far from the dicision boundary (far case) and the other where mislabeled data are concentrated near the dicision boundary (near case). Figure 3(a) shows the far case,

while Figure 3(b) shows the near case, where the decision boundary is shown by the solid line.

The number of training and test data are 1000 each. Iteration number of the weak learner, which is referred to as round (number), is 1000.



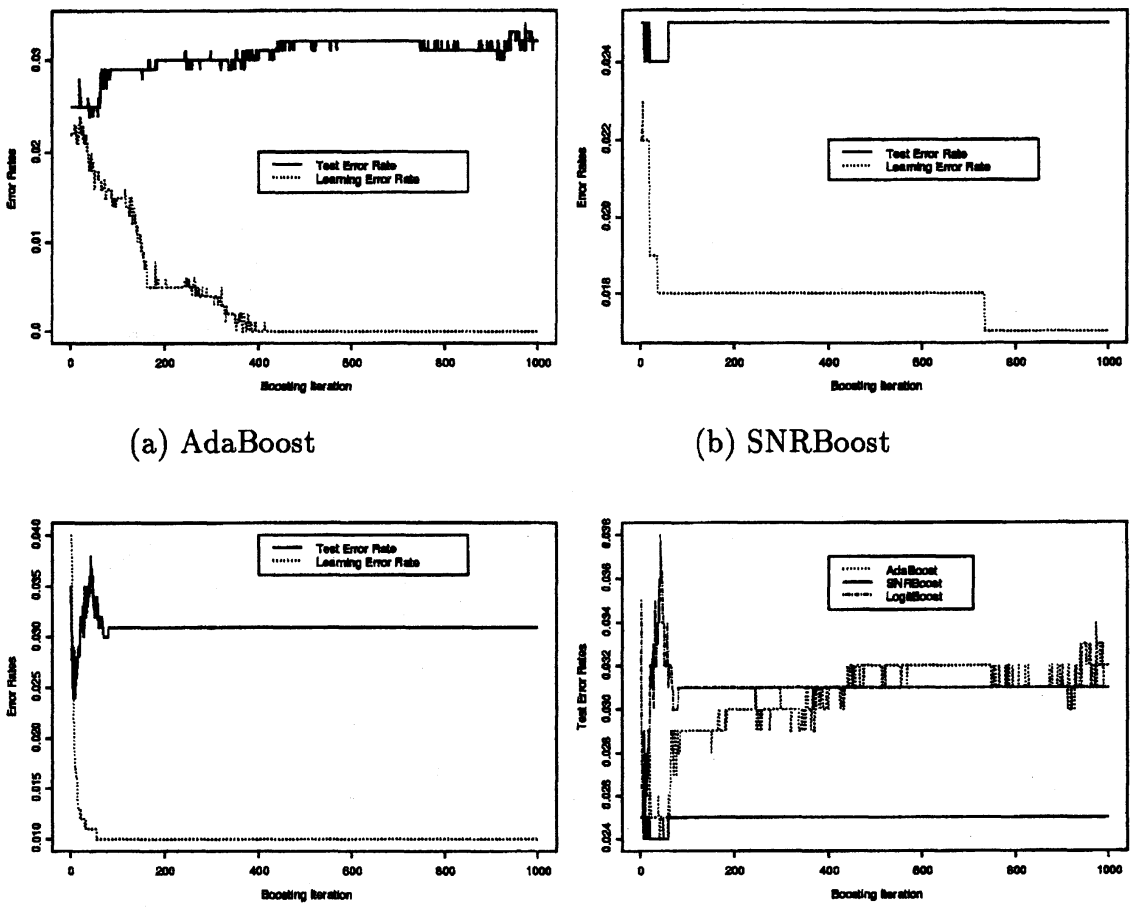
(a) AdaBoost (b) SNRBoost

(c) LogitBoost (d) Three methods

Figure 4: Comparison of three boosting methods in 2% mislabeled data far from decision boundary. (a) Learning and test error rates of AdaBoost. (b) Learning and test error rates of SNRBoost. (c) Learning and test error rates of LogitBoost. (d) Test error rates of three boosting methods.

We plot in Figures 4 the learning (error minimization) curves for 2% mislabeled data in far case, and in Figures 5 that for 2% mislabeled data in near case. In each figure, (a) shows AdaBoost learning and test error rates, (b) learning and test error rates of the proposed method, (c) learning and test error rates of the LogitBoost, and (d) comparison of test error rates of all the three boosting methods.

In Figures 4(a), AdaBoost learning error rate levels off into 0% around at 940 round (boosting iteration), and test error rate falls into 0.034% at 1000 round. In



(a) AdaBoost (b) SNRBoost

(c) LogitBoost (d) Three methods

Figure 5: Comparison of three boosting methods in 2% mislabeled data near decision boundary. (a) Learning and test error rates of AdaBoost. (b) Learning and test error rates of SNRBoost. (c) Learning and test error rates of LogitBoost. (d) Test error rates of three Boosting Methods.

Figure 4(b), SNRBoost learning error rate falls into 0.02% and test error rate into 0.026%. In Figure 4(c), LogitBoost learning error rate falls into 0.02% and test error rate into 0.025%. These facts dipict that SNRBoost and LogitBoost both succeeded in lowering test error rate. In Figure 4(d), the test error rate of SNRBoost is superior to AdaBoost and it exhibits equal performance to LogitBoost.

In Figures 5, the peformance of three boosting method for 2% mislabeled data in near case is shown. In Figure 5(a), AdaBoost learning error rate levels off into 0% around at 400 round and test error rate is fluctuating around 0.03%. In Figure 5(b), SNRBoost learning error rate falls into 0.017% at 1000 round and test error rate into 0.025%. In Figure 5(c), LogitBoost learning error rate falls into 0.01% and test error rate into 0.03%. From learning error rates of all the three boosting methods, we may observe that the mislabeled data near the dicision boundary is easier to classify than the mislabeled data far from the decision boundary. In Figure 5(d), test error rate of SNRBoost is superior to the other two boosting methods. The performance of LogitBoost is not as good as that in the far case. This may be due to the fact that the mislabeled data near the dicision boundary is easier to classify. In summary, SNRBoost exhibits a higher capability for generalization.

5. Conclusion

We developed a new, robust boosting method against mislabeled, noisy data. The new formulation uses the misclassification loss function, i.e., zero-one loss function. In deriving the algorithm, Stochastic Noise Reaction technique (Koda and Okano, 2000) is used to approximate the derivative of the zero-one loss function. Performance of the proposed method is compared with that of AdaBoost and LogitBoost through intensive numerical experiments. The proposed method is robust compared to the other two boosting methods especialy in the mislabeled data near dicision boundary. Even for mislabed data far from decision boundary, the proposed method exhibits the similar performance with that of LogitBoost.

Acknowledges

We thank Hiroyuki Okano for useful comments on an earlier draft. The work of M. Koda and H.Suzuki is supported in part by a Grant-in-Aid for Scientific Research of the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

Breiman, L. (1998): Combining Predictors. Technical Report, Statistics Department, University of California, Berkeley.

Dacol,D.K. and Rabitz, H. (1984): Sensitivity analysis of stochastic kinetic models. *J Math Phys*, **25**, 2716-2717.

Duda, R.O and Hart, P. E. and Stork,D. G.(2001): Pattern Classification, John Wiley & Sons, New York, NY.

Fournie, E. Lasry, J.M, Lebuchoux J, Lions, P.L, Touzi, N. (2001): Applications of Malliavin calculus to monte carlo methods in finance. *Finance Stochast.*, **5**, 201-236.

Freund, Y. (1995): Boosting a weak learning algorithm by majority. *Information and Computation*, **121** (2), 256-285.

Freund, Y. and Schapire, R. E. (1997): A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, **55** (1), 119-139.

Friedman, J. (2001): Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, **29** (5), 1189-1232.

Friedman, J., Hastie, T., and Tibshirani, R. (2000): Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, **28** (2), 337-407.

Grove, A. and Schuurmans, D. (1998): Boosting in the limit: Maximizing the margin of learned ensembles. *Proc. 15th National Conference on Artifical Intelligence*, 692-699.

Hastie, T., Tibshirani, R. and Friedman, J. (2001): The Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer, New York, NY.

Koda, M. and Okano, H. (2000): A new stochastic learning algorithm for neural networks. *Journal of the Operations Research Society of Japan*, **43** (4), 469-485.

Mason, L., Bartlett, P. L., and Baxter, J. (2000): Improved generalization through explicit optimization of margins. *Machine Learning*, **38** (3), 243-255.

Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. (2000): Functional gradient techniques for combining hypotheses. In A.J.Smola, P.L.Bartlet, B.Scholköpt, and D. Schuurmans, editors, *Advances in Large Margin Classifiers,* pages 221-246. MIT Press, Cambridge, MA, 2000.

Novikov EA. (1965): Functionals and the random-force method in turbulence theoty. *Sov Phys JETP*, **20**, 1290-1294.

Okano, H. and Koda, M. (2003): An optimization algorithm based on stochastic sensitivity analysis for noisy objective landscapes. *Reliability Engineering and System Safety*, **79** (2), 245-252.

Quinlan, J. R. (1996): Boosting first-order learning. *Proc. 7th International Workshop on Algorithmic Learning Theory*, **1160**, 143-155.

Rätsch, G., Onoda, T., and Müller, K.-R. (2001): Soft margin for AdaBoost. *Machine Learning*, **42** (3), 287-320.

Schapire, R. E. (1990): The strength of weak learnability. *Machine Learning*, **5** (2), 197-227.